本页总览

API简介

概述

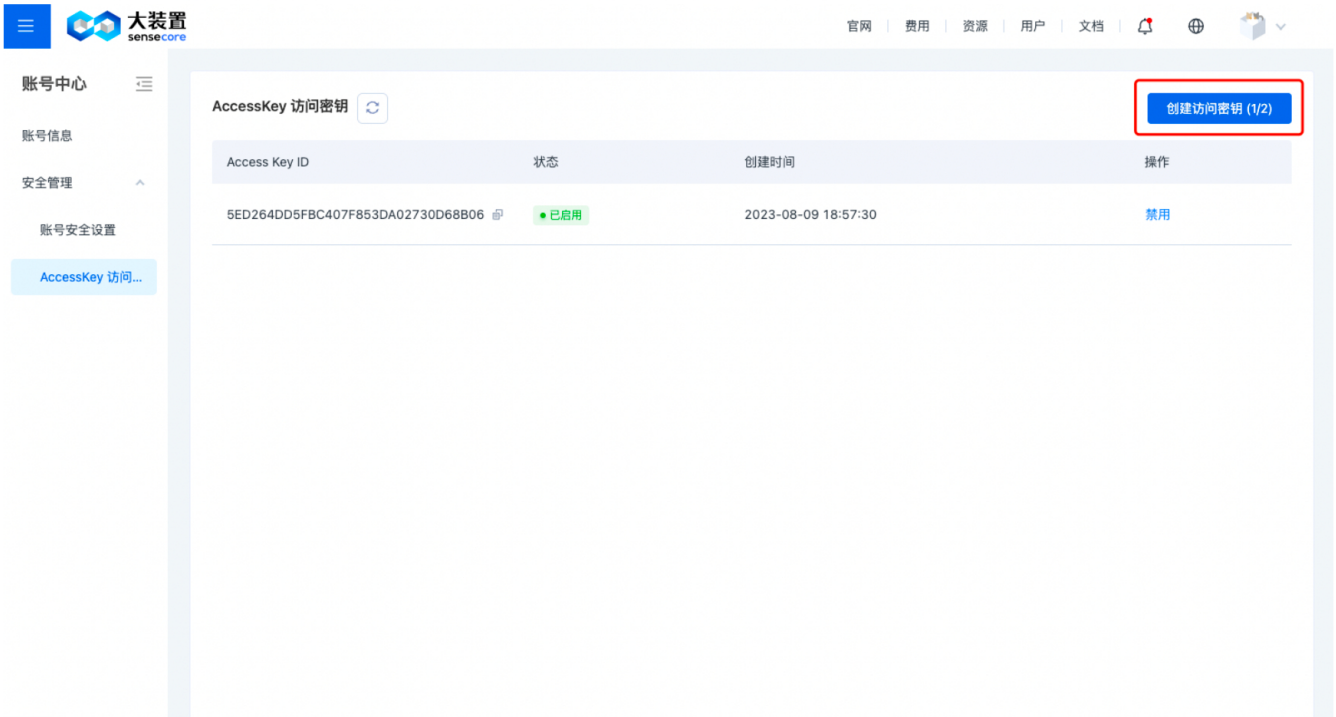
OpenAPI是SenseCore对外提供统一规范产品API服务的入口，为机构、服务商以及广大开发者提供统一标准、统一流程的API调用服务。
OpenAPI提供了REST（Representational State Transfer）风格API，支持您通过HTTPS请求调用，它使用统一的接口和标准的 HTTP 方法来进行资源的创建、读取、更新和删除操作。

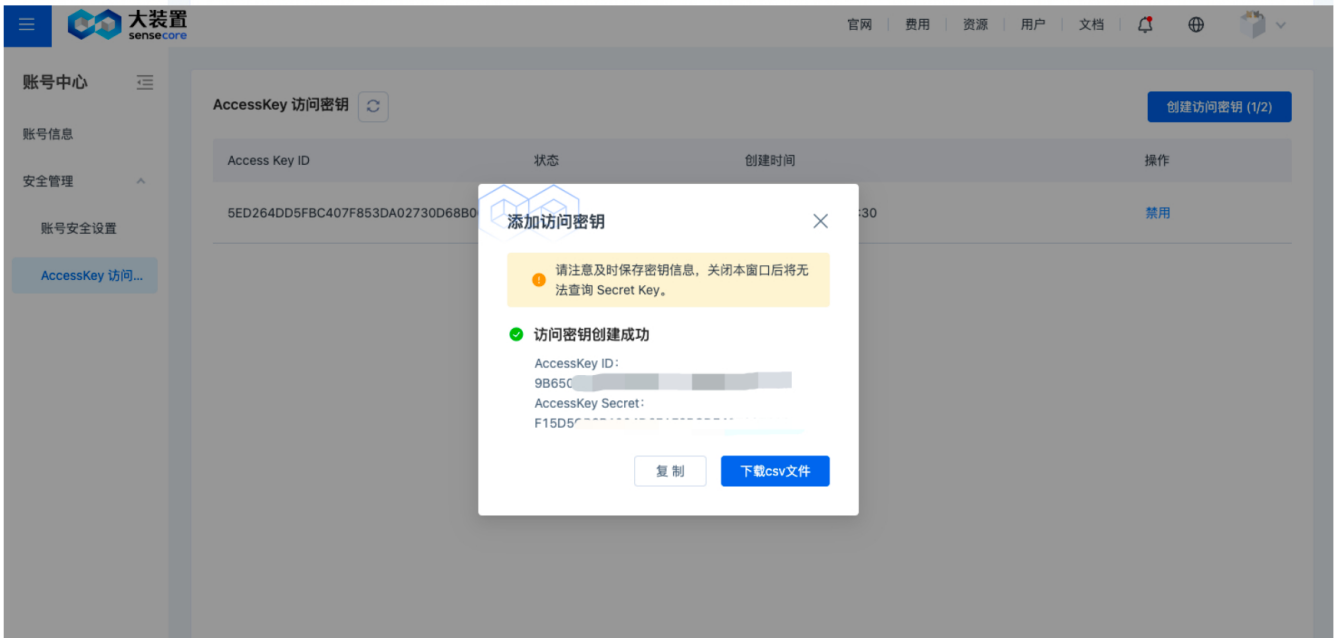
使用指南

API使用

第一步：创建AccessKey

1. 登录用户控制台，在头像-下拉菜单中点击 **AccessKey** 访问秘钥，进入账号中心-AccessKey 访问秘钥页。





2. 在AccessKey 访问秘钥页中，点击**创建访问秘钥**，创建一对属于当前账号的访问秘钥。秘钥对由 AccessKey ID 和 AccessKey Secret 组成，每个用户最多支持创建两对密钥对，创建后请注意即时保存 AccessKey Secret 信息。

第二步：查看API文档

1. 在顶导航中，点击文档，进入帮助中心。
2. 在文档中心中，选择并展开需要查看的产品服务名称，点击 API 参考查看相关 API 说明文档。

第三步：依据接口文档实现业务逻辑

您也可以点击接口文档中的调试接口按钮，进入 API 中心进行相关接口调试。

API调试

第一步：复制 Bearer 令牌

1. 在顶导航-头像下拉菜单中点击账号安全设置。
2. 点击复制按钮，复制 Bearer 令牌。

第二步：使用 Bearer 令牌进行 API 调试

1. 将复制好的 Bearer令牌粘贴到 API 调试页中的 Beaer Token 填写框中。
2. 开始进行 API 调试。

访问端点

详情请查看各个产品的OpenAPI说明文档

如何调用API

请求内容

请求URI

请求URI由如下部分组成。

{scheme} :// {Endpoint} / {resource-path} ? {query-string}

尽管请求URI包含在请求消息头中，但大多数语言或框架都要求您从请求消息中单独传递它，所以在此单独强调。

- **scheme**：表示用于传输请求的协议，当前所有API均采用**HTTPS**协议。
- **Endpoint**：指定承载REST服务端点的服务器域名或IP，不同服务不同区域的Endpoint不同

- **resource-path**：资源路径，也即API访问路径。
- **query-string**：查询参数，是可选部分，并不是每个API都有查询参数。查询参数前面需要带一个“？”，形式为“参数名=参数取值”，例如“limit=10”，表示查询不超过10条数据。

请求方法

HTTP请求方法（也称为操作或动词），它告诉服务你正在请求什么类型的操作。

- **GET**：请求服务器返回指定资源。
- **PUT**：请求服务器更新指定资源。
- **POST**：请求服务器新增资源或执行特殊操作。
- **DELETE**：请求服务器删除指定资源，如删除对象等。
- **HEAD**：请求服务器资源头部。
- **PATCH**：请求服务器更新资源的部分内容。当资源不存在的时候，PATCH可能会去创建一个新的资源。

请求消息头

附加请求头字段，如指定的URI和HTTP方法所要求的字段。例如定义消息体类型的请求头“Content-Type”，请求鉴权信息等。

如下公共消息头需要添加到请求中。

- X-Date: 该请求生成时的时间和日期。
- Authorization：该请求的认证信息 请求消息体

请求消息体

请求消息体通常以结构化格式发出，与请求消息头中Content-type对应，传递除请求消息头之外的内容。若请求消息体中参数支持中文，则中文字符必须为UTF-8编码。

每个接口的请求消息体内容不同，也并不是每个接口都需要有请求消息体（或者说消息体为空），GET、DELETE操作类型的接口就不需要消息体，消息体具体内容需要根据具体接口而定。

认证鉴权

访问凭证

Access Key（访问密钥）是一种用于身份验证和授权的凭据，用于访问和使用各种服务和API。Access Key 通常由用户在用户控制台-安全中心生成和管理。

Access Key 由两部分组成：Access Key ID（访问密钥标识符）和 Secret Access Key（访问密钥密钥）。Access Key ID 是用于标识访问密钥的唯一标识符，而 Secret Access Key 则是用于对请求进行签名和身份验证的机密密钥。

使用 Access Key 进行身份验证可以提供一定的安全性，因为只有持有正确的 Access Key 才能通过身份验证并获得访问权限。Access Key 通常需要妥善保管，避免泄露给未经授权的人员或应用程序。

在使用某些服务或API时，需要在请求中包含 Access Key 相关的信息，以证明身份并获得授权。具体的使用方法和要求取决于所使用的服务或API的提供商。通常，Access Key 相关的信息需要作为请求的一部分，可以通过请求头、请求参数或请求体的形式进行传递。

请求签名

平台提供了安全的**签名校验**方式，API访问中需要生成Http Authorization Header, 只有通过签名校验的请求才能到达业务后端。

```
Authorization: hmac accesskey="{accesskey}", algorithm="{algorithm}", headers="{headers}", signature="{s
```

参数名称	类型	说明
accesskey	string	用户的access key id
algorithm	string	hmac-sha256
headers	string	参与签名计算的header, 当前为x-date 和request-line
signature	string	请求签名

signature	string	请求签名

计算签名

对HTTP请求进行规范并取得请求的哈希值后，将其与签名算法、签名时间一起组成待签名字符串。

```
StringToSign =
    "x-data: {X-Data}" + \n +
    {RequestLine}
```

伪代码中参数说明如下。

- **X-Date**：请求时间戳。
- 格式参考https://datatracker.ietf.org/doc/html/rfc2616#section-3.3
- **RequestLine**：https://datatracker.ietf.org/doc/html/rfc2616#section-5.1

将SK（Secret Access Key）和创建的待签字符串作为加密哈希函数的输入，计算签名，将二进制值转换为base64表示形式。伪代码如下：

```
signature = Base64Encode(HMAC(Secret Access Key, StringToSign))
```

Secret Access Key	签名密钥
string to sign	创建的待签字符串

例子

AccessKey ID: 9eb0a32f-09c6-48da-8feb-34806dd60bdc

AccessKey Secret: secret

X-Date: Thu, 22 Jun 2017 17:15:21 GMT

request-line: GET /requests HTTP/1.1

```
StringToSign = "x-data: {X-Data}" + \n +{RequestLine}
             = "x-date: Thu, 22 Jun 2017 17:15:21 GMT\nGET /requests HTTP/1.1"
signature = base64(hmac_sha256("secret", StringToSign))
          = "IXlgb2baHcvPrV7a/C+hKS+E5oHIQXXyz4k4maWws50="
```

访问示例

```
package main

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/base64"
    "fmt"
    "net/http"
    "strings"
    "time"
)

func hmac_sha256(secret string, message string) string {
    key := []byte(secret)
```

```
    key := []byte(secret)
    h := hmac.New(sha256.New, key)
    h.Write([]byte(message))
    return base64.StdEncoding.EncodeToString(h.Sum(nil))
}

func main() {

    // information
    strAKID := "demo access key id" //替换为你的Access key
    strAKSecret := "demo access key secret" //替换为你的Access Secret
    strDateNow := time.Now().UTC().Format(http.TimeFormat)
    strAlgorithm := "hmac-sha256"
    strHeaders := "x-date request-line"

    // compute request signature
    strXDate := fmt.Sprintf("x-date: %s", strDateNow)
    strRequestLine := "GET /iam/idp/v1/users:getProfile HTTP/1.1"

    strSignContent := strings.Join([]string{strXDate, strRequestLine}, "\n")
    strSignature := hmac_sha256(strAKSecret, strSignContent)

    // generate Header Authorization
    strAuthorization := fmt.Sprintf(`hmac accesskey="%s", algorithm="%s", headers="%s", signature="%s"`,

    fmt.Println(strAuthorization)

    // send request
    strURL := "https://iam.sensecoreapi.dev/iam/idp/v1/users:getProfile"
    req, _ := http.NewRequest(http.MethodGet, strURL, nil)
    req.Header.Set("X-Date", strDateNow)
    req.Header.Set("Authorization", strAuthorization)

    client := http.Client{}

    _, err := client.Do(req)
        fmt.Println(err)

}
```

下一页： **创建数据集**